Assignment 1

Tokenisation :

Our aim is to tokenize i.e, split into tokens for a given corpus.It seems simple to tokenize text or separating words but it is not. From the given dataset we can see that it is not evenly spaced tokenized dataset made up of words.

Anime.txt

There are some short notations like do n't , he 's etc that has to be handled. Somewhere in the text occurrence of incomplete sentence e.g a ... is encountered , which has to be taken as single token instead of three separate ones.Square and curly brackets occur most frequently.Urls are also present in the text that comprises of both alphabets and numbers.So the tokenizer has been designed in such a way that handles such scenarios. Tokenizer here firstly replace all short notations of word to their original word and then the text is tokenized based on the special characters present in the English language.

Movies.txt

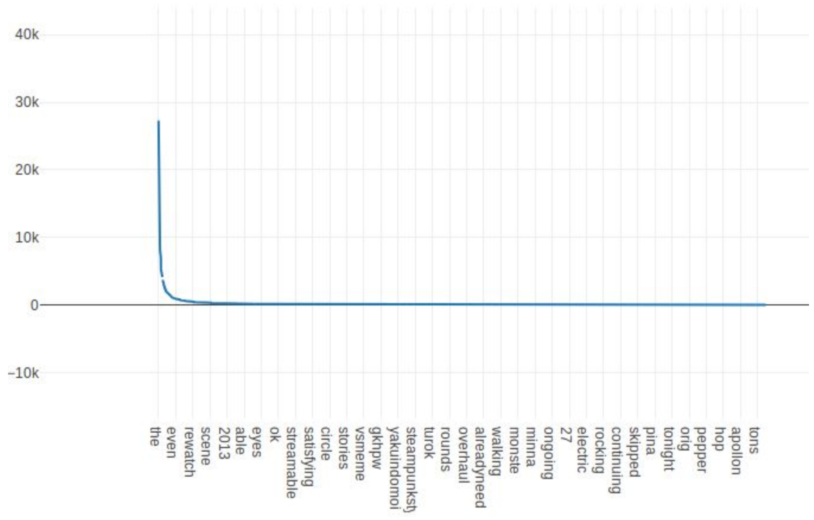
In the text , short notations of words needs to be handled . Somewhere in the text , there are &lt and &gt terms which are nothing but representation of less than and greater than symbol. Urls comprising of alphabets and numbers. Brackets occur irrespective of their usage. Incomplete sentences are also present . There are unicodes present that has to be handled. Tokenizer for this text is designed in the similar way as above , tokenizing based on the special characters present in the english language.

News.txt

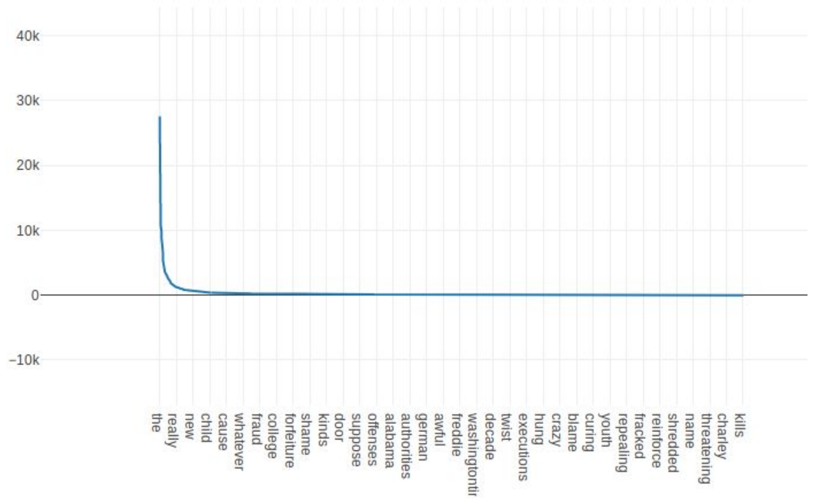
In this text is written in the same manner as been spoken without framing the sentence and hence because of that multiple incomplete sentences are encountered, various short notations, irregular spacing between words as well as sentence. Urls contains alphabets and numbers and there are hyphenated words that need to be considered as single token.Brackets occur unnecessarily in the text which signifies nothing. Some words like “mmmmm” , ”aaaaaaaaand” , ”lemmmegess” are also found in the text. At some places &gt which denoted greater than symbol is written.All these scenarios are handled while designing tokenizer that tokenizes text based on all special characters present in the language.

Language Modelling

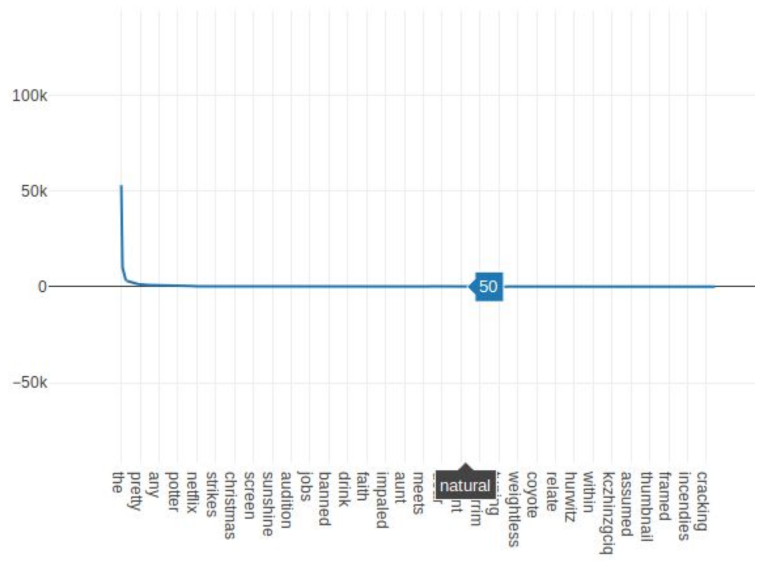
Ziph’s plot for unigrams



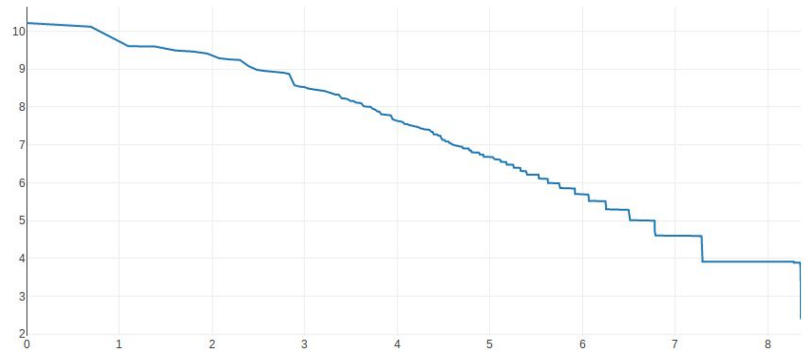
The above plot is frequency vs unigrams of text from anime.txt.



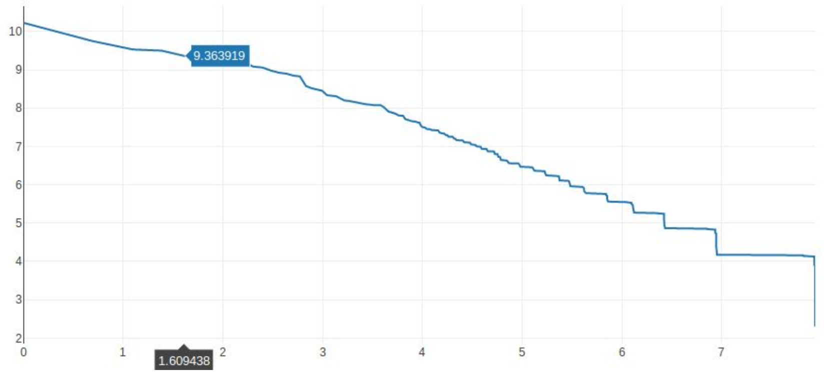
The above plotted graph is ziph's curve of unigrams text from news.txt file



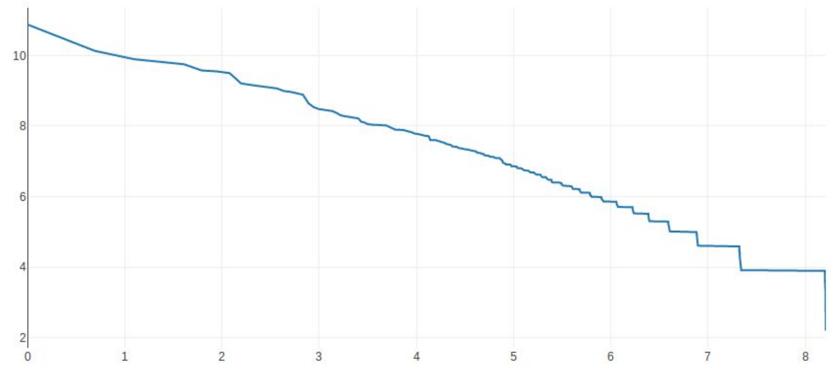
The above graph is Ziphs plot for unigrams of movies.txt



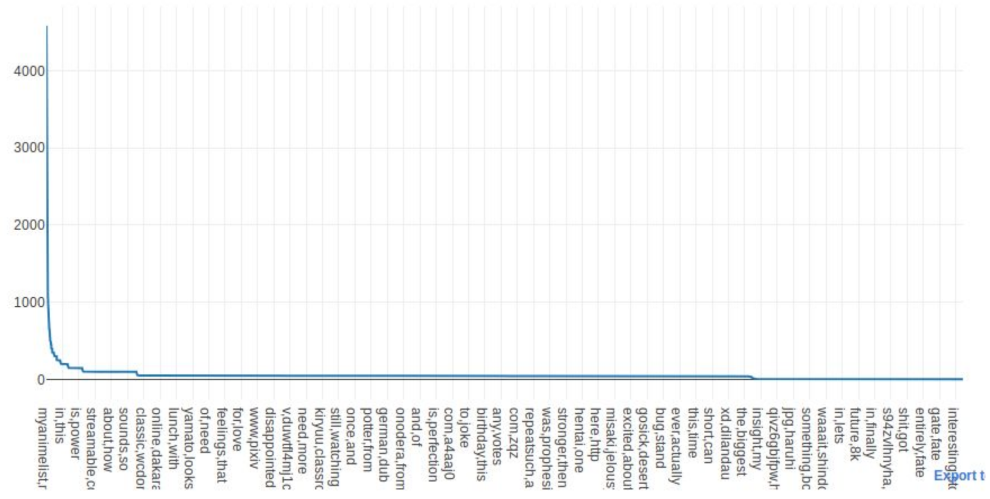
log(freq) vs log(ranks) graph of unigrams for anime.txt



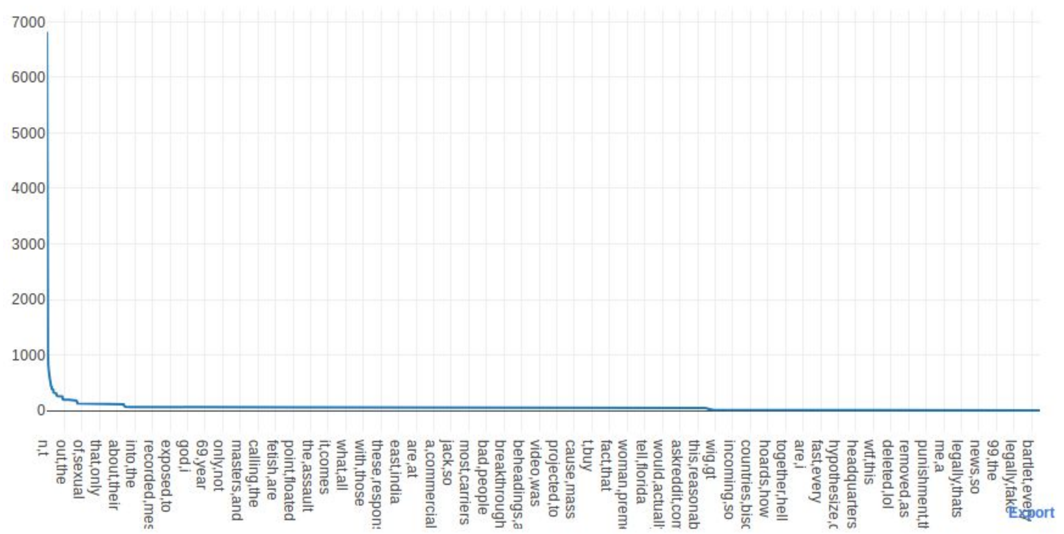
Log-log curve for unigrams of news.txt



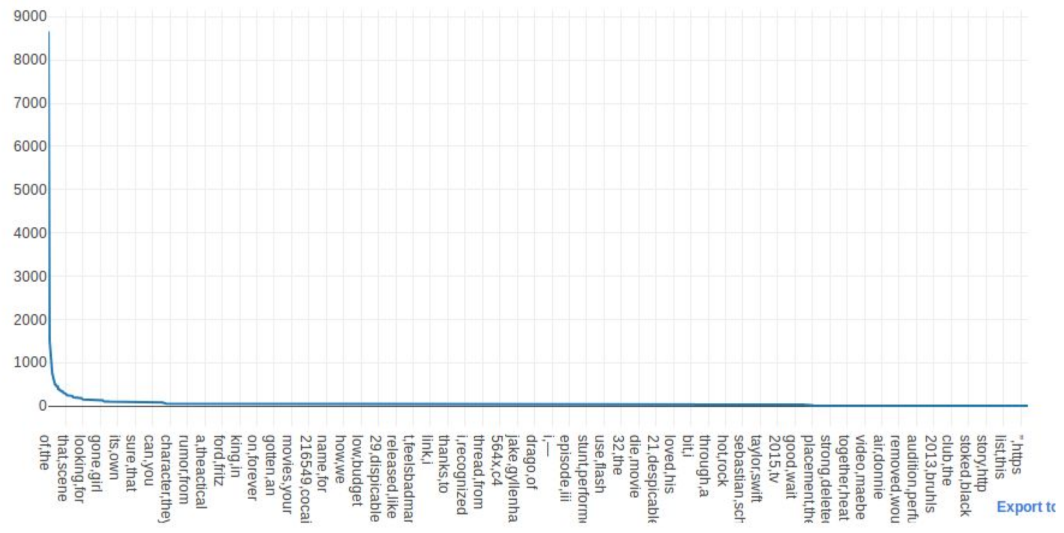
Log-log curve for unigrams of movies.txt



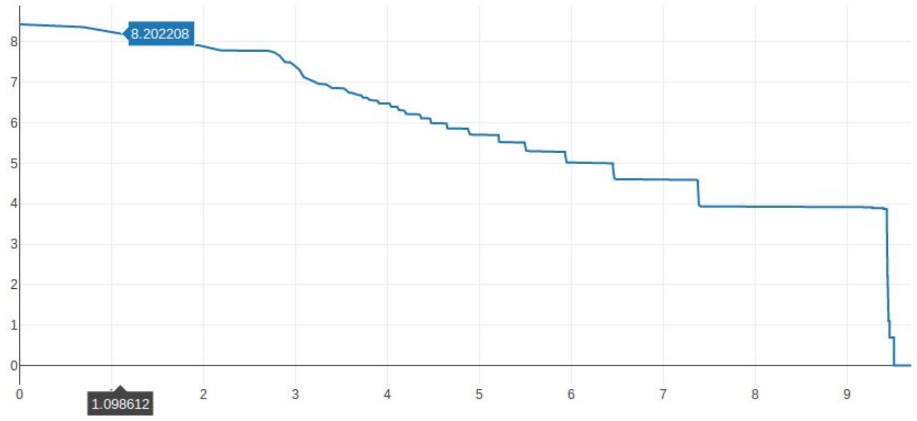
The above plot is ziphs plot for bigrams of anime.txt



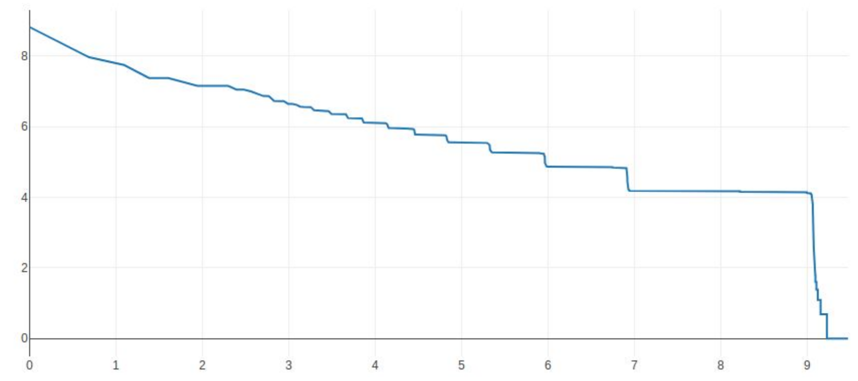
The above plot is ziphs plot for bigrams of news.txt



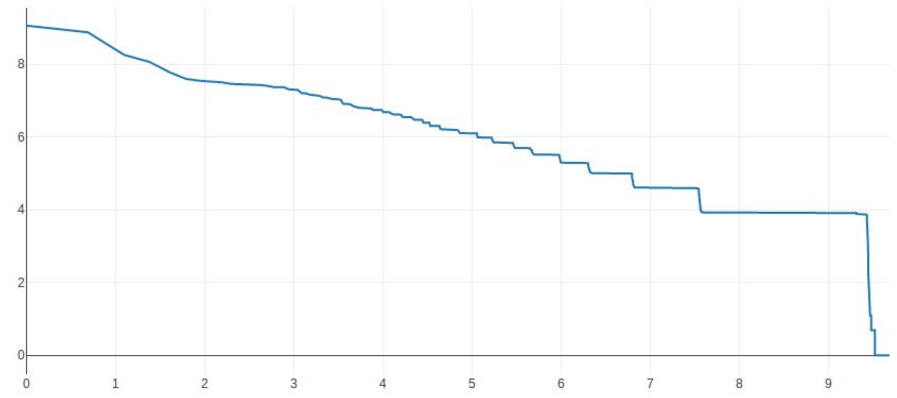
The above plot is ziphs plot for bigrams of movies.txt



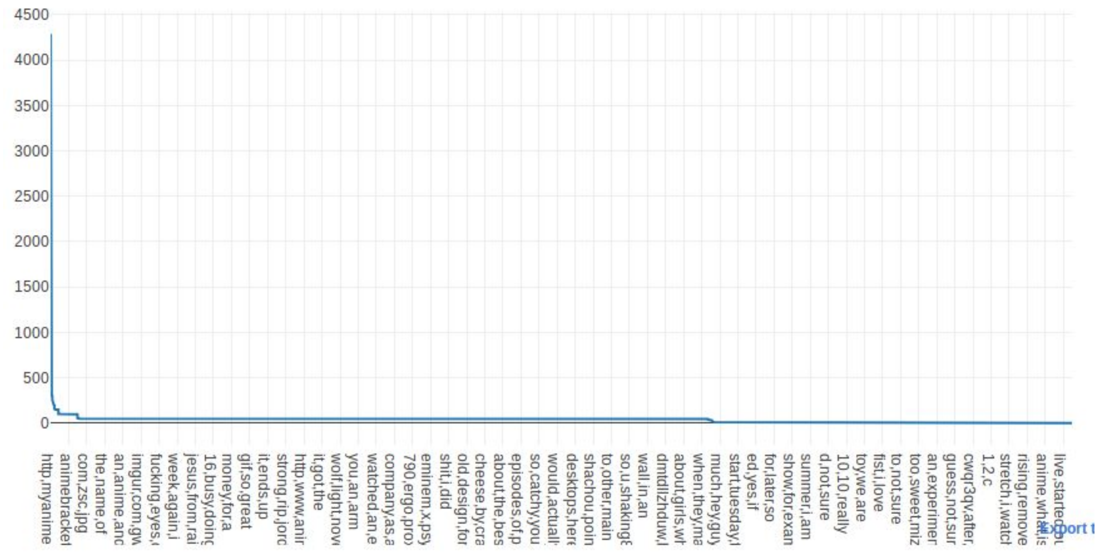
Log-log curve for bigrams of anime.txt



The above plot is log-log curve for bigrams of news.txt

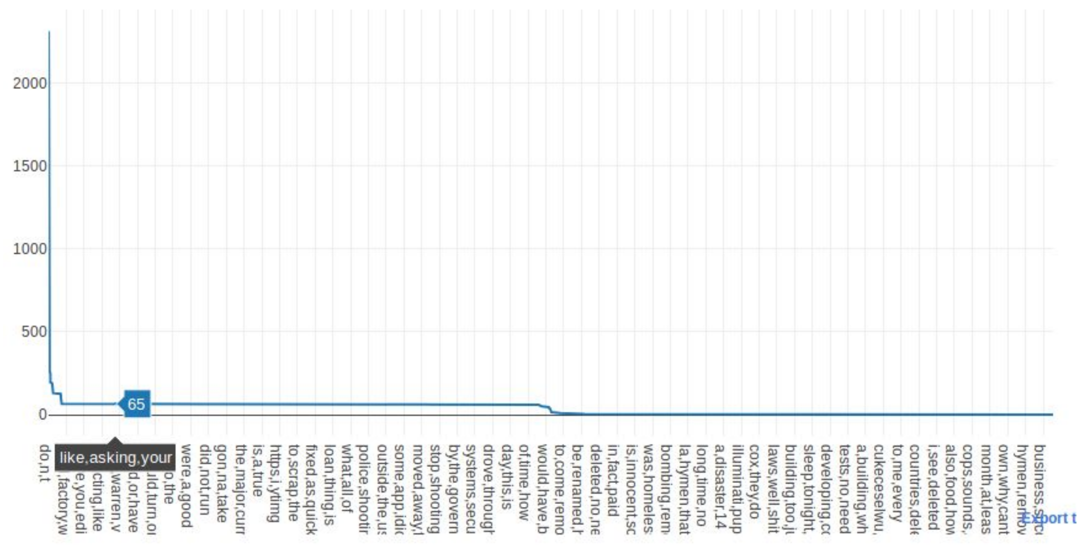


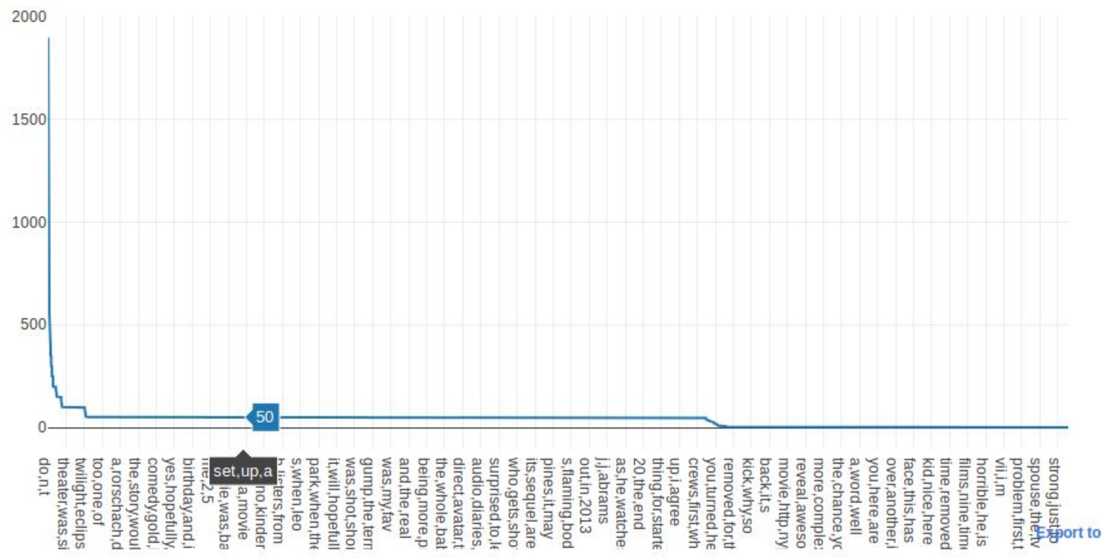
Plot of log-log curve for bigrams of movies.txt



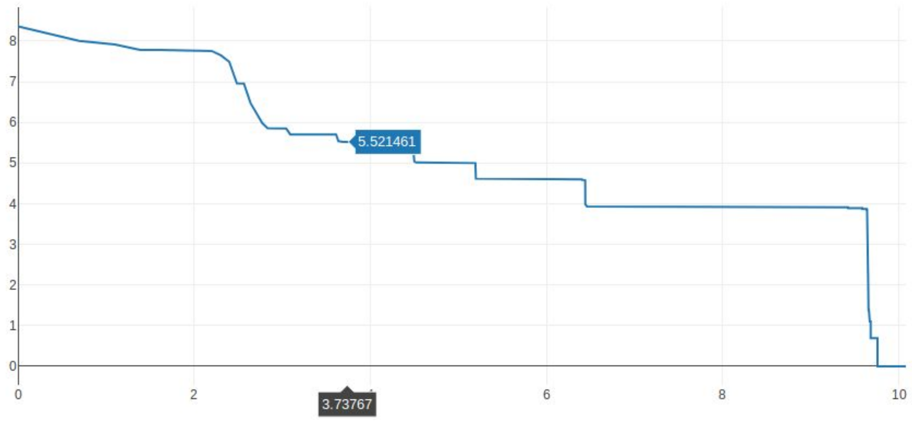
The above graph is ziphs plot for trigrams of anime.txt

The below is the ziphs curve for trigrams of news.txt

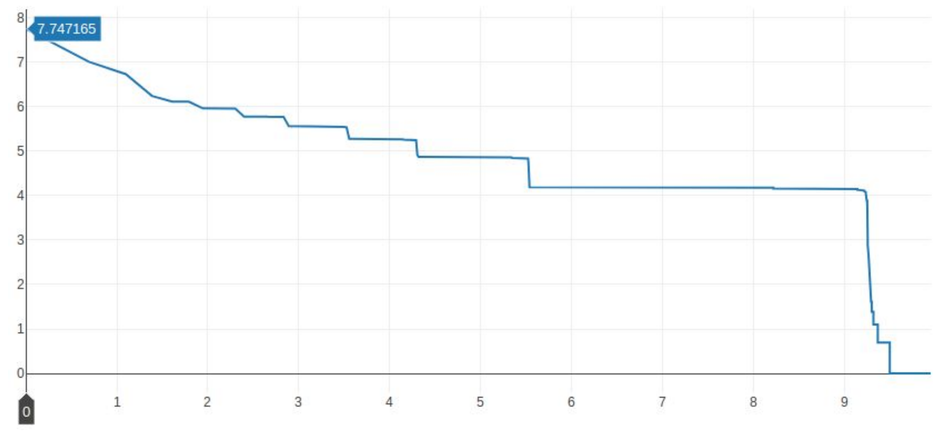




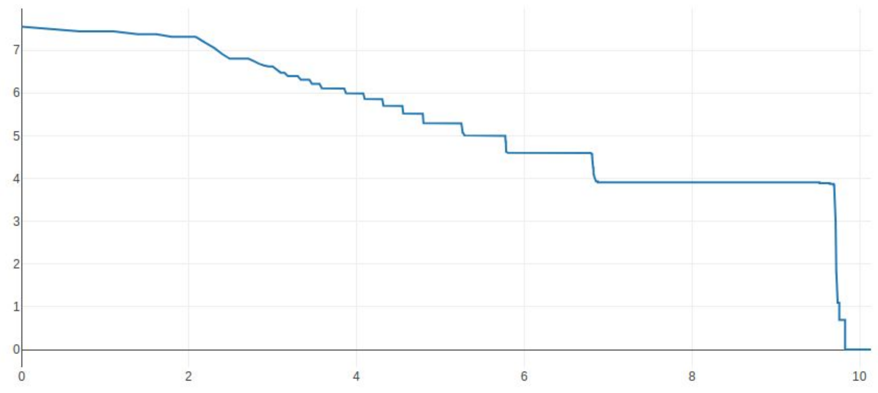
The above graph is ziphs curve for trigrams of movies.txt



The above plot is log(freq) vs log(ranks) for trigrams of anime.txt

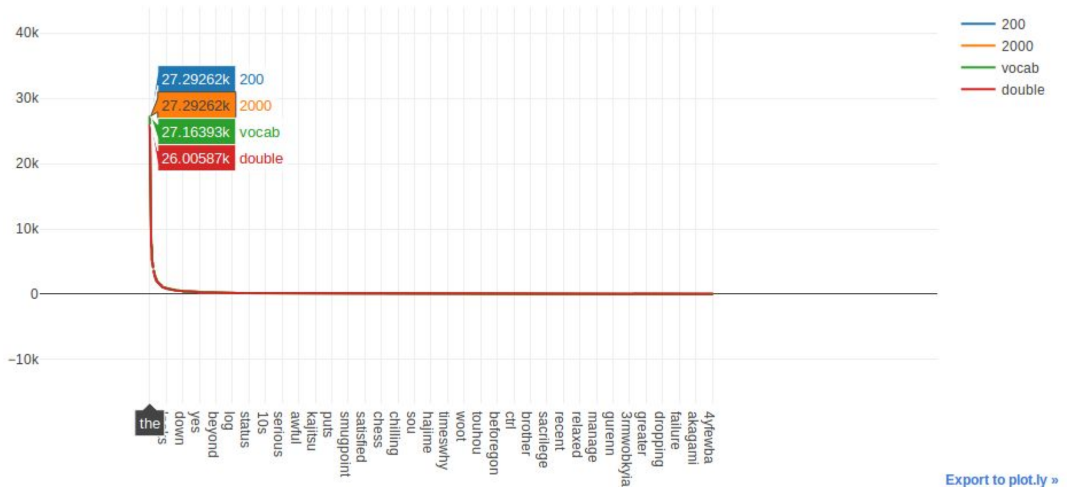


Above plot is log-log for trigrams of news.txt

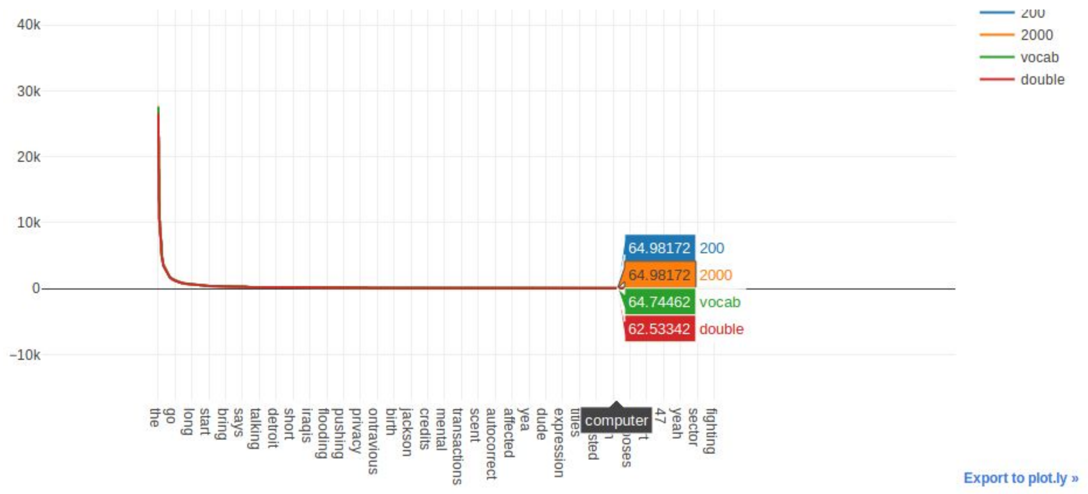


Log-log curve for trigrams of movies.txt

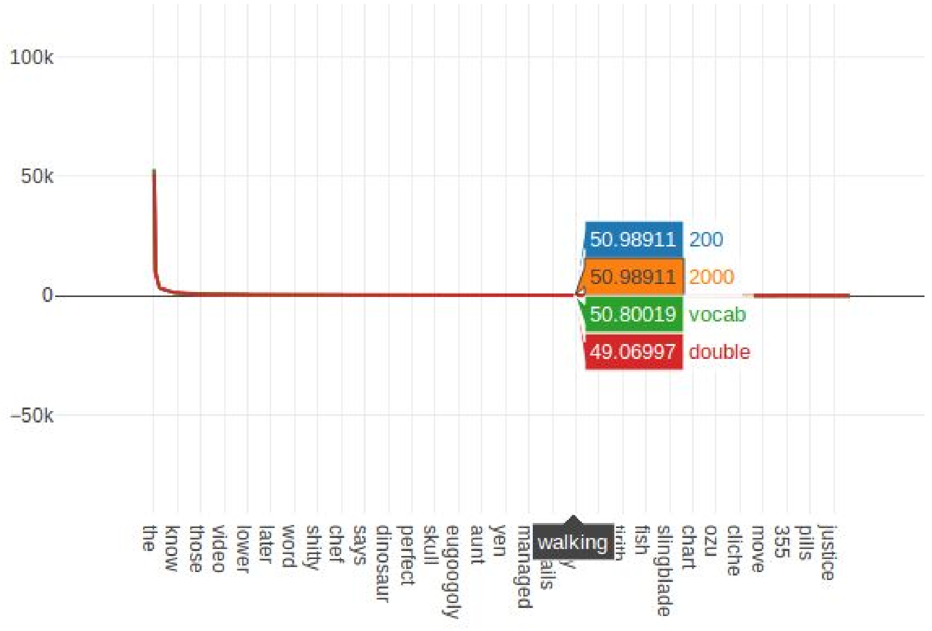
Laplace smoothing for unigrams of anime.txt for different values of V



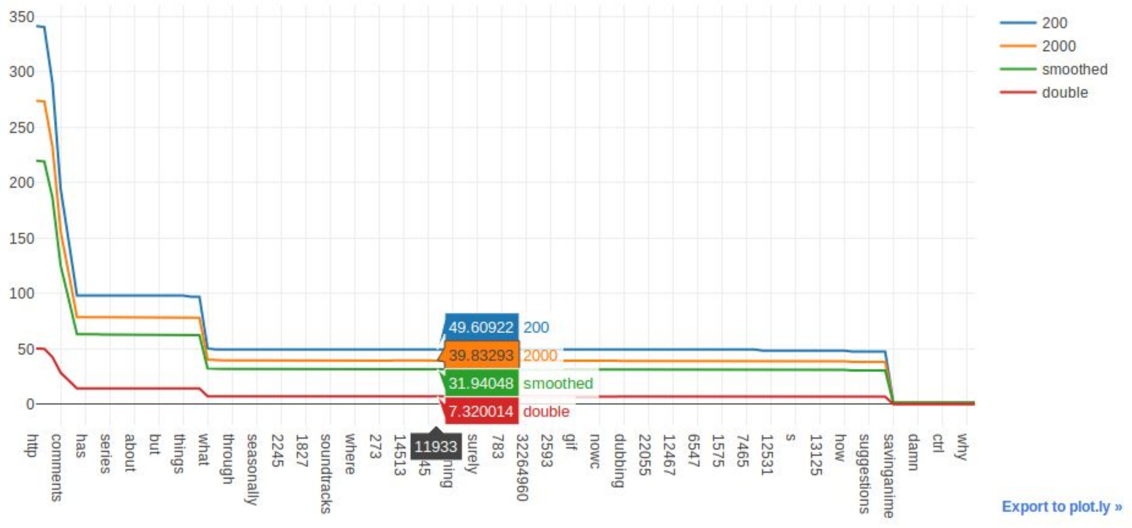
Laplace smoothing for unigrams of news.txt for different values of V



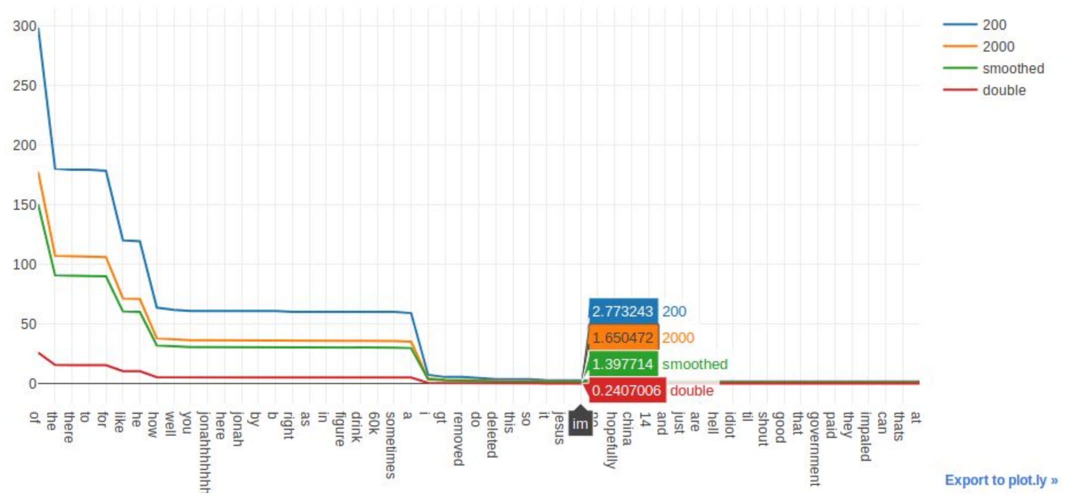
Laplace smoothing for unigrams of movies.txt for different values of V



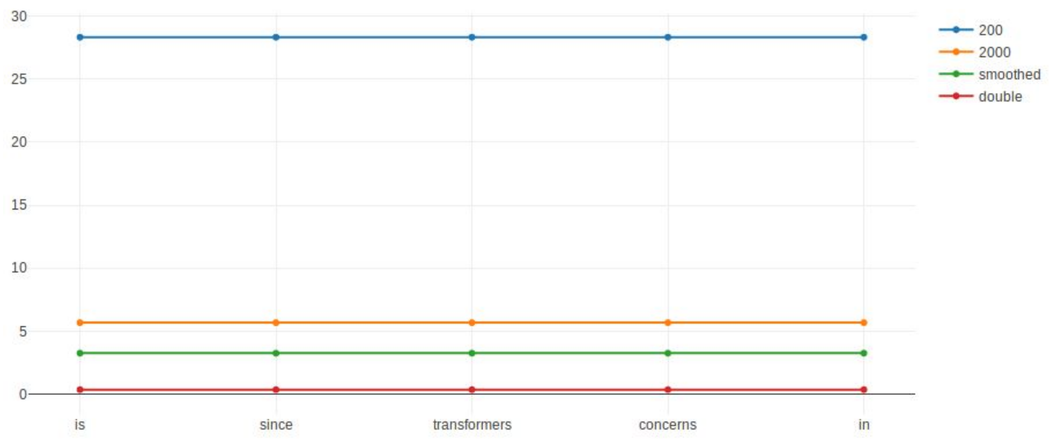
Laplace smoothing for bigrams which has w_{i-1} as 'anime' of anime.txt for different values of V



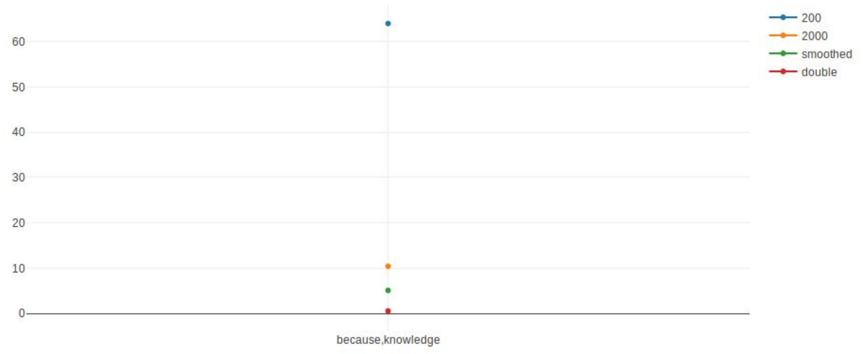
Laplace smoothing for bigrams which has w_{i-1} as 'out' of news.txt for different V values



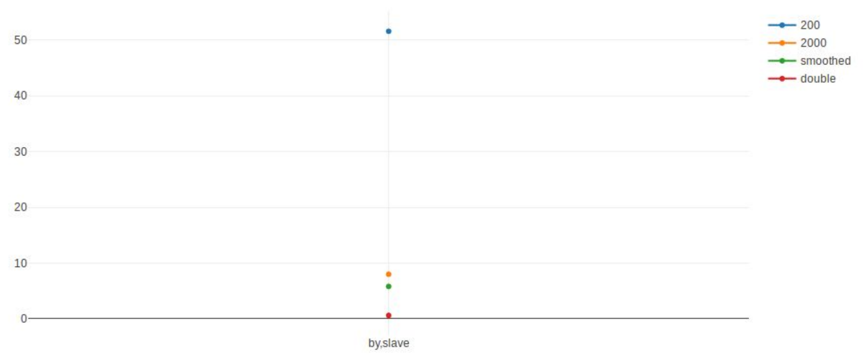
Laplace smoothing for bigrams which has w_{i-1} as 'budget' , of movies.txt for different V values

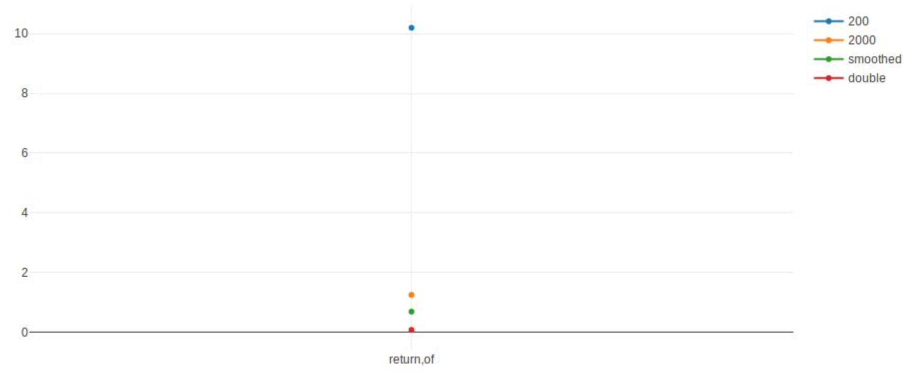


Laplace smoothing for trigrams which has bigram as ('because','knowledge') of anime.txt for different V values



Laplace smoothing for trigrams which has bigram as ('by','slave') of news.txt for different V values



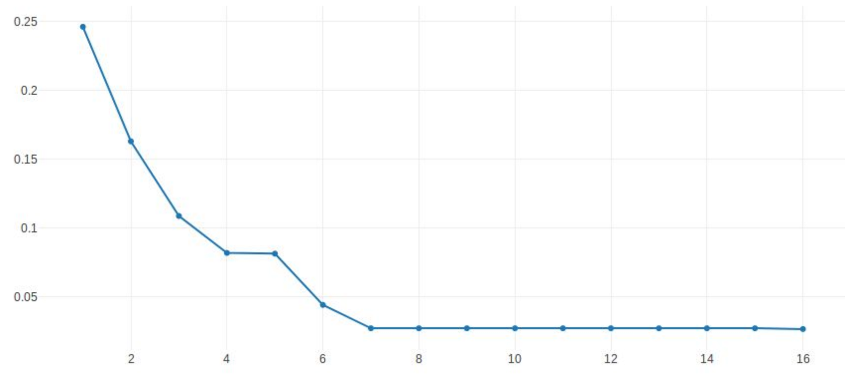


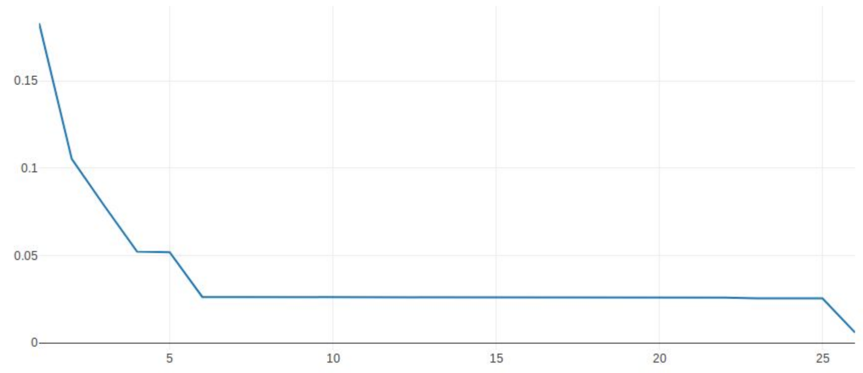
The above plot is laplace smoothing of trigrams for bigram as (‘return’,’of’) of movies.txt for different Values of V

For Witten Bell Smoothing on anime.txt , implemented it for given bigram (‘because’,’knowledge’) and computed its probability and count and compared it with the original count in the text. The original count is coming out to be 99 and that using witten bell is 97.81 and probability is 00655 while for trigram (‘because’,’knowledge’,’is’) original count is 99 and that for witten bell is 98.01 with probability 0.99. Similarly has done for news.txt by considering (‘by’,’slave’) as bigram and computing probability and count which comes out to be 0.0544 and 128.63 respectively and calculated the same for trigram(‘by’,’slave’,’labour’). For movies.txt , considering the bigram(‘return’,’of’) , probability and count comes out to be 0.712 and 249.28 respectively while the original count is 250 , similarly calculated for trigram.

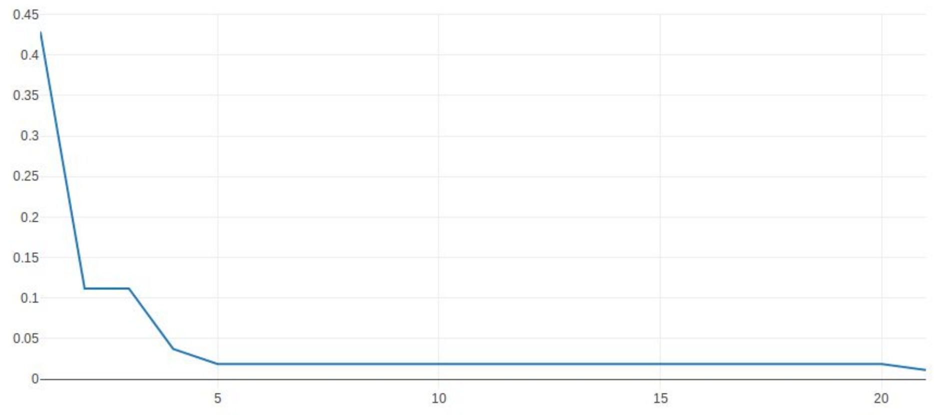
Kneser Ney

For anime.txt calculated the probabilities of all the bigrams that has w_{i-1} as ‘at’ and plotted the result with probabilities and ranks as axis.

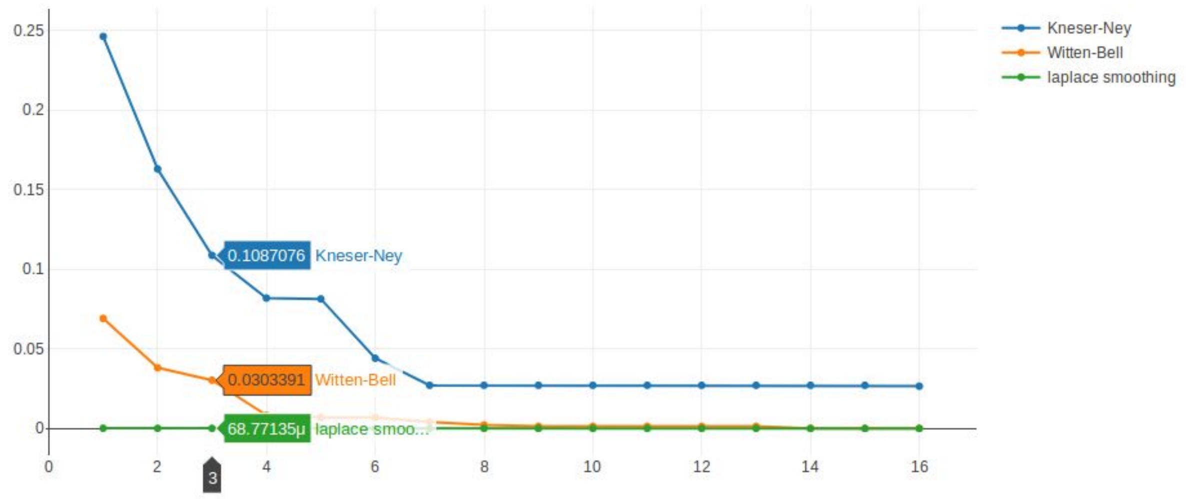




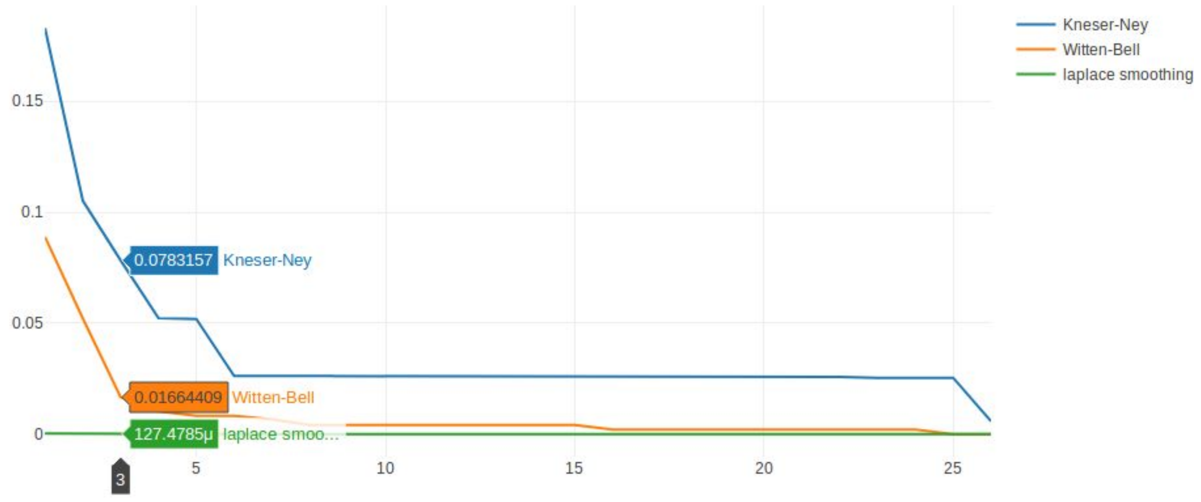
The similar approach is applied for text of news.txt and the result is plotted

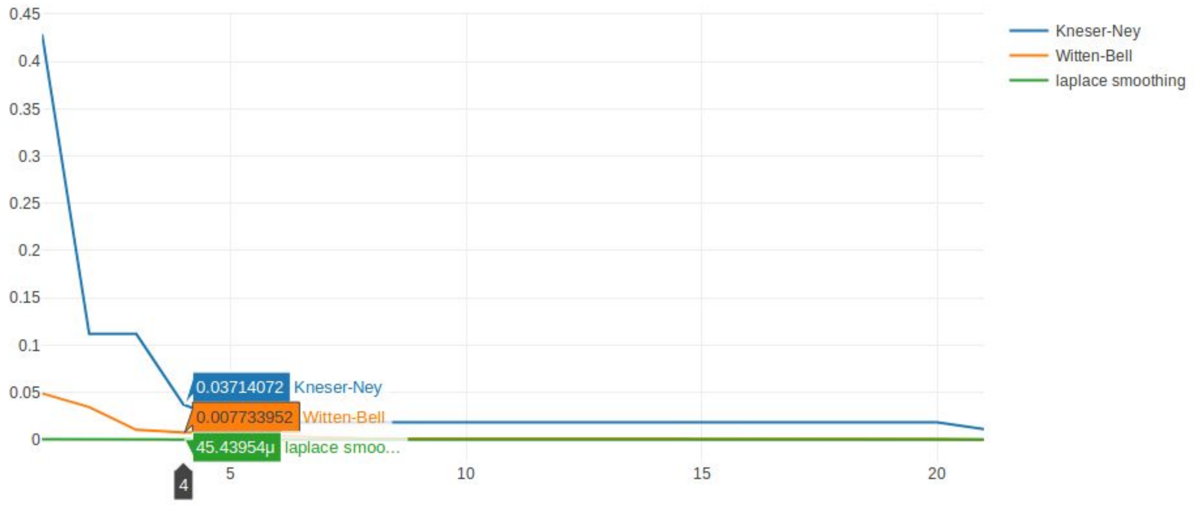


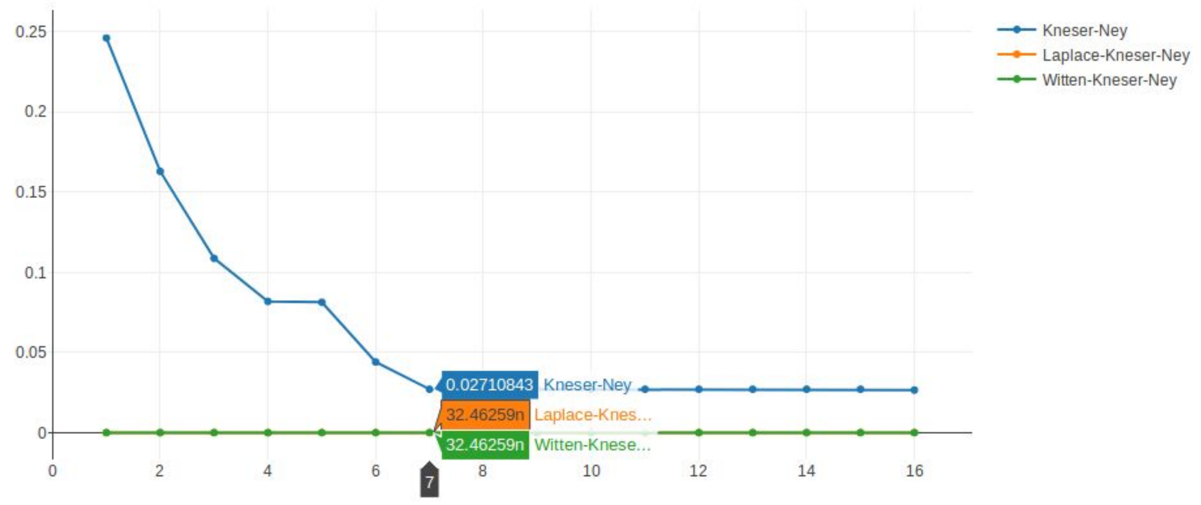
Similarly for movies.txt , where w_{i-1} still is ‘at’ .



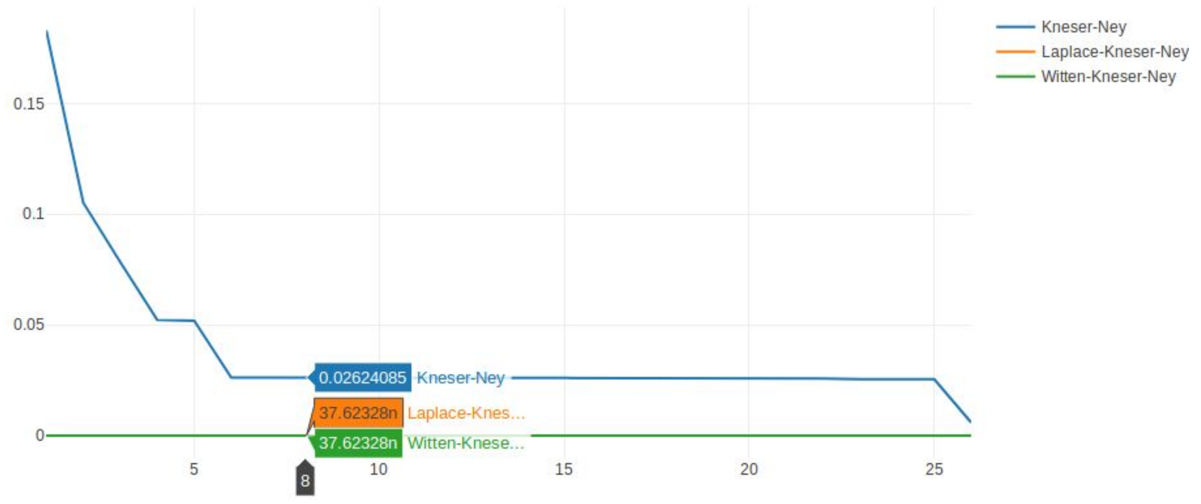
The above plot shows the comparison of probabilities obtained for bigrams which has w_{i-1} as 'at'. Plot is probabilities vs ranks.
The graph plotted below depicts the same , comparison amongst all smoothing techniques for the bigrams of news.txt with same w_{i-1} word.
The next graph also shows the same but for movies.txt file.

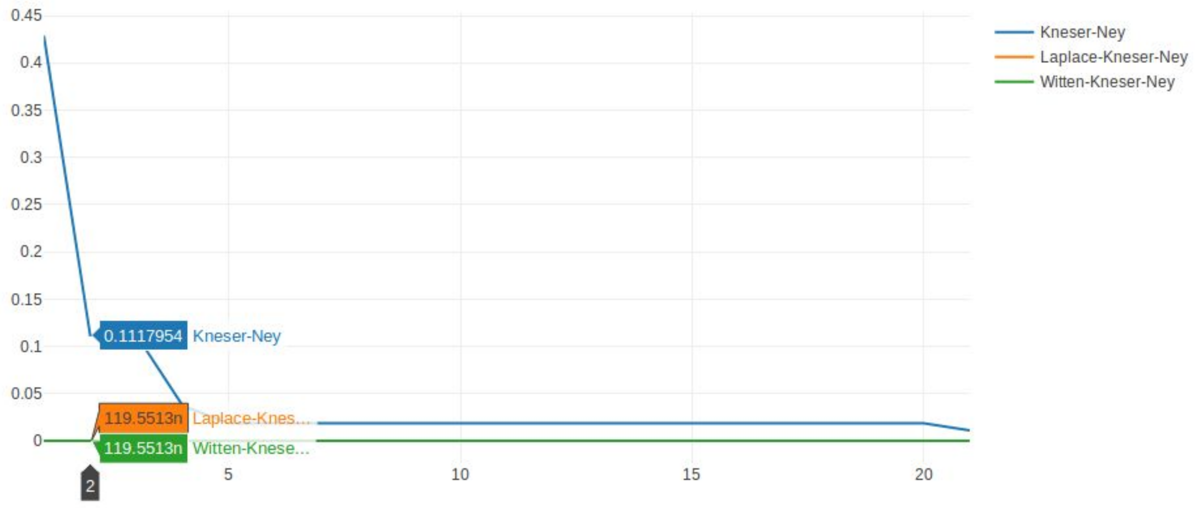






The above graph shows the comparison amongst modified kneser ney where the modification happened at the discounting step. The probabilities are computed for the bigrams of anime.txt which has w_{i-1} as ‘at’.The difference in probabilities is observed because in laplace , the probabilities of lower frequency words doesn’t change frequently while that significance change is observed in kneser ney because the second term enhances the result. In kneser ney both the terms balance each other and hence we get significance probabilities.
Witten Bell gives better result compared to laplace smoothing but the second term as said before increases the probability while this doesn’t happen in witten bell.



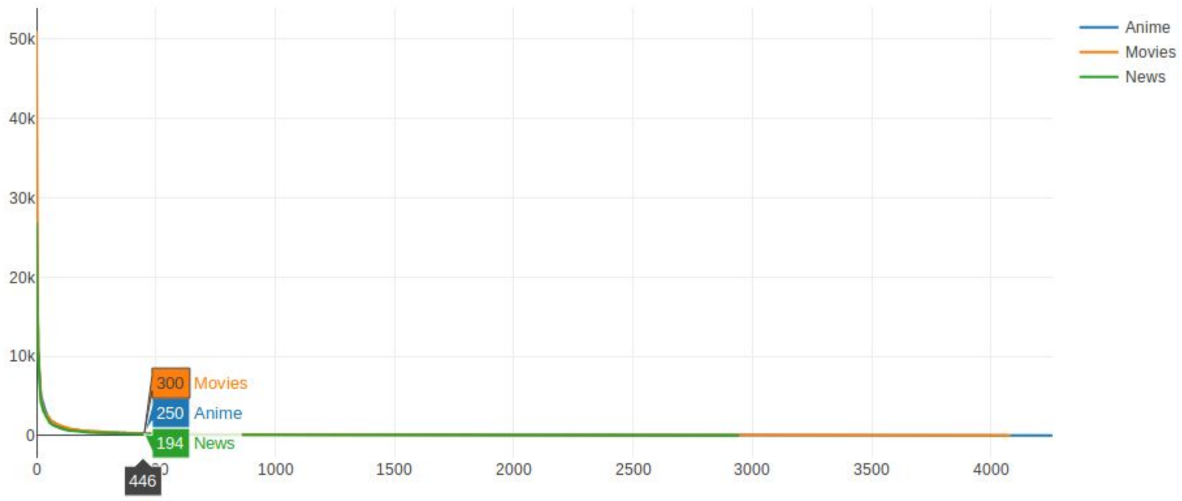


The above two graphs are of bigrams which has w_{i-1} as ‘at’ of news.txt and movies.txt respectively. The difference in plots has occurred due the same reason as stated above.

In text generation , considered the highly frequent unigram and then using that unigram created bigram which has higher kneser ney probability then after that used this bigram to create trigram which has highest probability and then used the bigrams thereafter to get the entire text. The stopping condition for this is the number of words to generate in the text.

Naive Bayes

Plot of ziphs curve for unigrams of three sources



Some words of anime are not present in annotated file but then too when given these words in text for classification, it classified accurately to anime class.This shows the Naive Bayes is classifying precisely.